Biomedical Optics EXPRESS

Automatic analysis of bioresorbable vascular scaffolds in intravascular optical coherence tomography images

YIHUI CAO,[1,2,3,7] QINHUA JIN,[4,7] YIFENG LU,[1,3] JING JING,[4] YUNDAI CHEN,[4,*] QINYE YIN,[2] XIANJING QIN,[5,6] JIANAN LI,[1] RUI ZHU,[1] AND WEI ZHAO[1]

[1] *State Key Laboratory of Transient Optics and Photonics, Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an 710119, Shaanxi, China*
[2] *School of the Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China*
[3] *University of Chinese Academy of Sciences, Beijing 100049, China*
[4] *Department of Cardiology, Chinese PLA General Hospital, Beijing, China*
[5] *Department of Aerospace Biodynamics, Fourth Military Medical University, Xi'an 710032, Shaanxi, China*
[6] *Xidian University, Xi'an 710071, Shaanxi, China*
[7] *These authors contributed equally to this work.*
[*] *cyundai@vip.163.com*

Abstract: The bioresorbable vascular scaffold (BVS) is a new generation of bioresorbable scaffold (BRS) for the treatment of coronary artery disease. A potential challenge of BVS is malapposition, which may possibly lead to late stent thrombosis. It is therefore important to conduct malapposition analysis right after stenting. Since an intravascular optical coherence tomography (IVOCT) image sequence contains thousands of BVS struts, manual analysis is labor intensive and time consuming. Computer-based automatic analysis is an alternative, but faces some difficulties due to the interference of blood artifacts and the uncertainty of the struts number, position and size. In this paper, we propose a novel framework for a struts malapposition analysis that breaks down the problem into two steps. Firstly, struts are detected by a cascade classifier trained by AdaBoost and a region of interest (ROI) is determined for each strut to completely contain it. Then, strut boundaries are segmented within ROIs through dynamic programming. Based on the segmentation result, malapposition analysis is conducted automatically. Tested on 7 pullbacks labeled by an expert, our method correctly detected 91.5% of 5821 BVS struts with 12.1% false positives. The average segmentation Dice coefficient for correctly detected struts was 0.81. The time consumption for a pullback is 15 *sec* on average. We conclude that our method is accurate and efficient for BVS strut detection and segmentation, and enables automatic BVS malapposition analysis in IVOCT images.

© 2018 Optical Society of America under the terms of the OSA Open Access Publishing Agreement

OCIS codes: (100.6950) Tomographic image processing; (170.4500) Optical coherence tomography; (100.2960) Image analysis.

References and links

1. S. C. Smith, A. Collins, R. Ferrari, D. R. Holmes, S. Logstrup, D. V. McGhie, J. Ralston, R. L. Sacco, H. Stam, K. Taubert, D. A. Wood, and W. A. Zoghb, "Our time: a call to save preventable death from cardiovascular disease (heart disease and stroke)," Eur. Heart J. **33**(23), 2910–2916 (2012).
2. T. Takagi, G. Stankovic, L. Finci, K. Toutouzas, A. Chieffo, V. Spanos, F. Liistro, C. Briguori, N. Corvaja, R. Albero, G. Sivieri, R. Paloschi, M. C. Di, and A. Colombo, "Results and long-term predictors of adverse clinical events after elective percutaneous interventions on unprotected left main coronary artery," Circulation **106**(6), 698–702 (2002).
3. J. W. Moses, M. B. Leon, J. J. Popma, P. J. Fitzgerald, D. R. Holmes, C. O'shaughnessy, R. P. Caputo, D. J. Kereiakes, D. O. Williams, P. S. Teirstein, J. L. Jaeger, and R. E. Kuntz, "Sirolimus-eluting stents versus standard stents in patients with stenosis in a native coronary artery," N. Engl. J. Med. **349**(14), 1315–1323 (2003).

#325099 https://doi.org/10.1364/BOE.9.002495
Journal © 2018 Received 6 Mar 2018; revised 19 Apr 2018; accepted 24 Apr 2018; published 1 May 2018

4. J.-i. Kotani, M. Awata, S. Nanto, M. Uematsu, F. Oshima, H. Minamiguchi, G. S. Mintz, and S. Nagata, "Incomplete neointimal coverage of sirolimus-eluting stents," J. Am. Coll. Cardiol. **47**(10), 2108–2111 (2006).
5. G. J. Tearney, E. Regar, T. Akasaka, T. Adriaenssens, P. Barlis, H. G. Bezerra, B. Bouma, N. Bruining, J. M. Cho, and S. Chowdhary, "Consensus standards for acquisition, measurement, and reporting of intravascular optical coherence tomography studies : A report from the international working group for intravascular optical coherence tomography standardization and validation," J. Am. Coll. Cardiol. **59**(12), 1058–1072 (2012).
6. A. J. Brown, L. M. Mccormick, D. M. Braganza, M. R. Bennett, S. P. Hoole, and N. E. West, "Expansion and malapposition characteristics after bioresorbable vascular scaffold implantation," Catheter. Cardio. Inte. **84**(1), 37–45 (2014).
7. N. Gonzalo, P. W. Serruys, N. Piazza, and E. Regar, "Optical coherence tomography (oct) in secondary revascularisation: stent and graft assessment," EuroIntervention **5**(Suppl D), D93–100 (2009).
8. A. Elnakib, G. Gimel'farb, J. S. Suri, and A. Elbaz, *Multi Modality State-of-the-art Medical Image Segmentation and Registration Methodologies* (Springer Science & Business Media), Chap. 1.
9. A. Wang, J. Eggermont, N. Dekker, H. M. Garcia-Garcia, R. Pawar, J. H. Reiber, and J. Dijkstra, "Automatic stent strut detection in intravascular optical coherence tomographic pullback runs," Int. J. Cardiovas. Imag. **29**(1), 29–38 (2013).
10. G. J. Ughi, T. Adriaenssens, K. Onsea, P. Kayaert, C. Dubois, P. Sinnaeve, M. Coosemans, W. Desmet, and J. D'hooge, "Automatic segmentation of in-vivo intra-coronary optical coherence tomography images to assess stent strut apposition and coverage," Int. J. Cardiovas. Imag. **28**(2), 229–241 (2012).
11. G. T. Bonnema, K. O. Cardinal, S. K. Williams, and J. K. Barton, "An automatic algorithm for detecting stent endothelialization from volumetric optical coherence tomography datasets," Phys. Med. Biol. **53**(12), 3083 (2008).
12. S. Gurmeric, G. Isguder, S. Carlier, and G. Unal, "A new 3-d automated computational method to evaluate in-stent neointimal hyperplasia in in-vivo intravascular optical coherence tomography pullbacks," in *Proceedings of Medical Image Computing and Computer-Assisted Intervention* (Springer, 2009) pp. 776–785.
13. C. Xu, J. M. Schmitt, T. Akasaka, T. Kubo, and K. Huang, "Automatic detection of stent struts with thick neointimal growth in intravascular optical coherence tomography image sequences," Phys. Med. Biol. **56**(20), 6665 (2011).
14. A. Wang, S. Nakatani, J. Eggermont, Y. Onuma, H. M. Garcia-Garcia, P. W. Serruys, J. H. Reiber, and J. Dijkstra, "Automatic detection of bioresorbable vascular scaffold struts in intravascular optical coherence tomography pullback runs," Biomed. Opt. Express **5**(10), 3589–3602 (2014).
15. K. R. Foster, R. Koprowski, and J. D. Skufca, "Machine learning, medical diagnosis, and biomedical engineering research - commentary," Biomed. Eng. Online **13**(1), 94 (2014).
16. P. Viola and M. Jones, "Robust real-time face detection," Int. J. Comput. Vision **57**(2), 137–154 (2004).
17. A. A. Amini, T. E. Weymouth, and R. C. Jain, "Using dynamic programming for solving variational problems in vision," IEEE T. Pattern. Anal. **12**(9), 855–867 (1990).
18. T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," IEEE T. Pattern. Anal. **24**(7), 971–987 (2002).
19. N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2005), pp. 886–893.
20. Y. Cao, Q. Jin, Y. Chen, Q. Yin, X. Qin, J. Li, R. Zhu, and W. Zhao, "Automatic side branch ostium detection and main vascular segmentation in intravascular optical coherence tomography images," IEEE J. Biomed. Health. (to be published).
21. Y. Cao, Q. Jin, Y. Chen, Q. Yin, X. Qin, J. Li, R. Zhu, and W. Zhao, "Automatic identification of side branch and main vascular measurements in intravascular optical coherence tomography images," in *Proceedings of IEEE International Symposium on Biomedical Imaging* (IEEE, 2017), pp. 608–611.
22. R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," Commun. ACM **15**(1), 11–15 (1972).
23. B. D. Gogas, V. Farooq, Y. Onuma, and P. W. Serruys, "The absorb bioresorbable vascular scaffold: an evolution or revolution in interventional cardiology," Hell. J. Cardiol. **53**(4), 301–309 (2012).
24. G. Rätsch, T. Onoda, and K. R. Müller, "Soft margins for adaboost," Mach. Learn. **42**(4), 287–320 (2001).

1. Introduction

Cardiovascular disease is the world's NO.1 cause of death in recent years, accounting for 17.3 million deaths per year, a number that is expected to grow to 23.6 million by 2030 [1]. Coronary stenting is the most common treatment modality for coronary artery disease. However, metallic stents face the potential risk of late stent thrombosis [2]. The drug-eluting stent (DES) is a later generation of stent design which can significantly reduce in-stent restenosis [3]. However, multiple risk factors become evident 3-6 months after DES implantation [4] and therefore lead to risks of late stent thrombosis in the long term. The third generation of stent is "Bioresorbable Scaffolds (BRS)", which is seen as one of the most promising type of scaffolds in the long term.

Absorb by Abbott (bioresorbable vascular scaffold, BVS) is one of the many scaffold types of BRS. Although BVS has been retracted from the market, studies on automatic analysis of BVS struts are of importance since they are potentially useful for the analysis of other types of BRS with similar features. One of the major studies of BVS is malapposition analysis. Malapposition is defined as incomplete strut apposition when axial distance between the abluminal surface of struts and the endoluminal surface of the vessel wall is larger than strut thickness (about 150 μm) [5], and is considered as a potential cause of late stent thrombosis [6].

Intravascular optical coherence tomography (IVOCT), due to its superior resolution, is currently the state-of-the-art imaging modality that enables BVS malapposition analysis [7]. In current practice, malapposition analysis in IVOCT images is mainly conducted qualitatively by human experts. which may face the problem of subjectivity and inaccuracy. Therefore, quantitative malapposition analysis is of important clinical value. However, current quantitative analysis is time consuming and labor intensive due to the large amounts of image frames in a pullback. Therefore, fast and quantitative malapposition analysis for BVS is highly desired and of great clinical value.

Since the location, size and number of BVS struts are uncertain and vary from image to image, it's difficult to directly segment each strut by traditional segmentation methods [8]. However, if the position of each strut is detected in advance, the segmentation problem can be resolved by traditional methods [8]. Therefore, it is important to accurately detect each strut before segmentation. Automatic stent detection is still at an early stage worldwide and most attention has been focused on metallic stent. Some previous studies have been proposed in recent years [9–13]. However, these methods have only utilized local features of metallic stent and can hardly be applied to BVS detection since metallic stent is quite different from BVS. To the best of our knowledge, the only published work for BVS detection is conducted by Wang et al. [14], whose method is mainly based on gray and gradient features and uses a large number of thresholds to detect BVS struts. However, since the intensity and contrast of IVOCT images vary, and there may exist some blood artifacts in the vessel, this method with fixed empirical parameters and thresholds is difficult to generalize in practice.

On the other hand, machine learning has been increasingly employed in medical image processing [15] due to its advantage of high robustness and generalization. Therefore in order to obtain better generalization, in this paper, a cascade of classifiers is trained based on Adaboost [16] and is then employed to detect BVS struts center position and region. After that, dynamic programming (DP) [17] is deployed to segment the boundary of each strut. Based on the segmentation result, malapposition analysis is conducted automatically. Given the above aspects, our method enables automatic BVS struts analysis in IVOCT images. In order to evaluate our method, we compared automatic and manual analysis results. Qualitative and quantitative validation showed that the presented method is efficient and robust.

The rest of the paper is organized as follows: the detailed method for BVS struts detection and segmentation is illustrated in section 2. The experimental setup and evaluation method are presented in section 3. The experimental results are presented and discussed in section 4. Conclusions are drawn in section 5.

2. Method

As is shown in Fig. 1, BVS struts are represented by bright rectangular boundaries with box-shape black cores inside. The goal is to detect and segment each strut to quantitatively evaluate malapposition. The workflow mainly contains three steps: 1) Struts detection. The main goal is to detect the location and size of BVS struts; 2) Struts segmentation. The main task is to segment the struts boundaries after detection; 3) Malapposition evaluation. With detection and segmentation results, malapposition evaluation is conducted for further analysis. In the following subsections, each step is described in more detail.

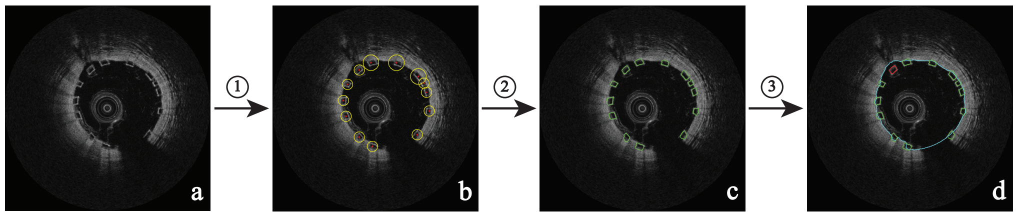


Fig. 1. Procedures of BVS struts analysis. Each step is: ① Struts detection; ② Struts segmentation; ③ Malapposition analysis;

2.1. *Struts detection*

In our method, machine learning is employed for the detection of struts position and size. A key point in machine learning is to decide what features to extract. In practice, there are many kinds of features to select such as Haar-like features [16], Local Binary Pattern (LBP) [18] and Histogram of Oriented Gradient (HOG) [19], among which Haar-like features have proven to be quite effective for line and edge detection, and could be computed in a short time based on the integral image. Therefore, Haar-like features are a potentially efficient feature type for BVS struts detection. On the other hand, AdaBoost [16] has shown great robustness, quickness and accuracy for object detection in the wild. Therefore in our method, a cascade classifier is trained by AdaBoost [16] based on Haar-like features and is then used for BVS struts detection. The main goal is to find the center position and size of each strut and determine a region of interest (ROI) that completely contains the strut. Details are shown in the following three parts.

2.1.1. Feature extraction

Fig. 2 visualizes some basic models of Haar-like features. The feature value is defined as the difference between the sum of the pixels within black and white regions. The model moves pixel by pixel throughout the whole image to compute the feature values at different position. It is then enlarged and again moves throughout the image to conduct the computation at different scales. In this way, the features at different positions and different scales are computed successively.

However, due to the wide range of scales and positions, the amount of Haar-like features can be extremely large. For example, a 24×24 image contains more than 70000 Haar-like features. It is therefore time consuming to directly compute all these feature values in the original image. Integral image provides an efficient approach to solve this problem. Specifically, an integral image $II(x, y)$ is the sum of intensities located in up-left region of the original image $I(x, y)$. In this way, the integral image $II(x, y)$ can be defined as:

$$II(x, y) = \sum_{x'=1}^{x} \sum_{y'=1}^{y} I(x', y') \tag{1}$$

Then, the sum of any given rectangular region $D(x_1, y_1, x_2, y_2)$ can be computed by:

$$D = II(x_1, y_1) + II(x_2, y_2) - II(x_1, y_2) - II(x_2, y_1) \tag{2}$$

where points $(x_1, y_1), (x_2, y_2), (x_1, y_2), (x_2, y_1)$ are the four vertexes of the rectangular region D. Thus the intensity sum of a rectangle is computed with only four lookups. From that, any Haar-like feature can be calculated with several lookups from the integral image in a short time.

Once all Haar-like features are computed, it is important to decide which feature to select for the classification of strut and non-strut. In our approach, Adaboost algorithm is applied to select the most effective Haar-like features for struts detection.

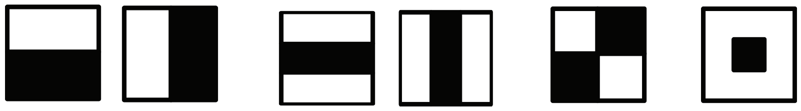


Fig. 2. Some basic models of Haar-like features.

2.1.2. Classifiers training

In our approach, the training mainly contains two parts: training strong classifiers by AdaBoost, and constructing the strong classifiers into a cascade. Each part is illustrated in detail as follows.

a. Multilayer-tree AdaBoost

The basic idea of AdaBoost is to combine different weak classifiers into a strong classifier that performs much better than any single weak one. The simplest weak classifier is a decision stump defined as:

$$h(f, \boldsymbol{x}, p, \theta) = \begin{cases} 1 & \text{if } pf > p\theta \\ -1 & \text{else} \end{cases} \tag{3}$$

where f is the feature value, $\boldsymbol{x}$ is the training sample, p represents polarity and θ is threshold. A total of N training samples are labeled as $(\boldsymbol{x}_1, y_1)$ ...$(\boldsymbol{x}_N, y_N)$ where $y_i = 1$ for positive samples and $y_i = -1$ for negative samples. Each sample is classified by h based on Eq.(3). If $h = y_i$, the sample is correctly classified. Otherwise, it is misclassified. For each feature, a threshold θ and a polarity p are found to minimize the error rate e defined as the weighted sum of misclassified samples. The feature with the minimal error rate is then chosen as the most distinguishable feature in the weak classifier. Fig. 3(a) shows the structure of a weak classifier based on a decision stump.

A single weak classifier contains only one feature and can hardly reach excellent performance on its own. In AdaBoost, this problem can be solved by combining different weak classifiers into a strong one. The training for the following weak classifiers is similar as described above, except that the weight distribution of training samples are changed iteratively. The key point is to increase the weight of the misclassified samples so that the training will focus more and more on hard examples. Once all weak classifiers are trained, they are combined into a strong one. In AdaBoost, a weight α_j is assigned to each weak classifier based on its error rate e. Specifically, the lower e is, the larger weight a weak classifier is assigned with. The strong classifier is boosted by the weighted weak classifiers. Fig. 3(b) shows a strong classifier consisting of J weak ones.

AdaBoost is very efficient for feature selection. However, due to the large number of Haar-like features, different weak classifiers may share severe redundancy and can't reach the optimal performance once combined together. In order to solve this problem, in our method, decision tree is selected as the weak classifier instead of a simple stump, as Fig. 3(c) shows. By maximizing the decrease in error rate Δe between parent and child node, the tree minimizes the redundancy among the features within a weak classifier. Specifically, the first node is trained by selecting the feature with the minimum error rate e. Then, the feature for the child node is selected to maximize the decrease in error rate. This process is conducted repeatedly until the maximum split is reached. In our method, the maximum split is set to three, as Fig. 3(c) shows.

The training of the strong classifier is similar to that illustrated in Fig. 3(b). The weight distribution of misclassified samples is increased iteratively. Once all weak classifiers are trained, they are assigned with different weights to construct a strong classifier. Traditionally, one weak classifier is assigned with one weight. However, this idea neglects the difference in classification ability of different nodes within a weak classifier. In our method, each weak classifier is assigned with four weights $\alpha_{j,1}$ to $\alpha_{j,4}$ corresponding to the four leaf nodes, as Fig. 3(d) shows.

The training process of decision tree-based AdaBoost is represented by pseudo code as follows:

Input:

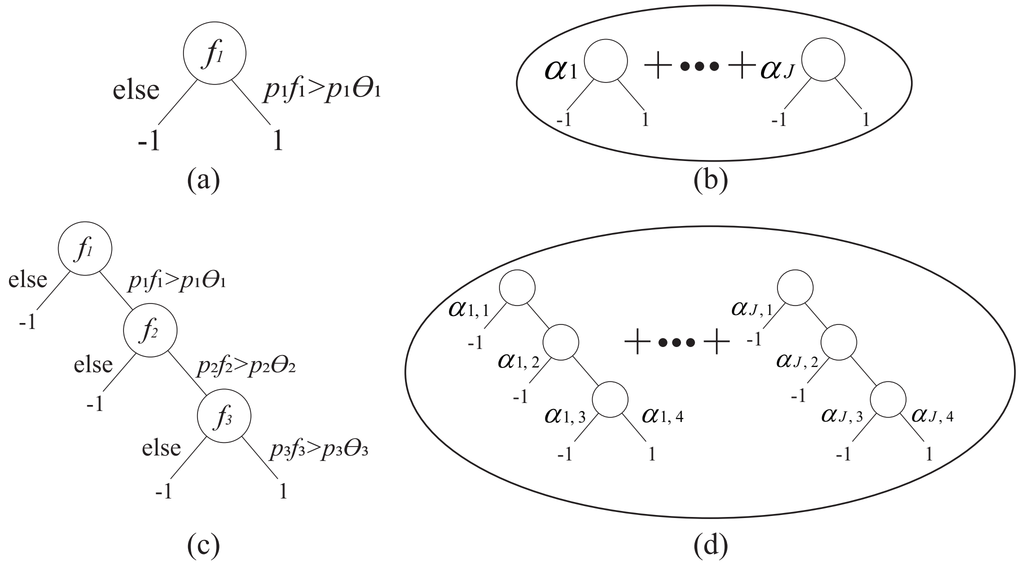


Fig. 3. The structure of a weak classifier built up by (a) a decision stump and (c) a decision tree. (b) and (d) are the strong classifier boosted by (a) and (c), respectively.

1) A set of training samples $(\boldsymbol{x}_1, y_1)$ ...$(\boldsymbol{x}_N, y_N)$, where $y_i = 1, -1$ for positive and negative samples respectively.
2) Number of desired iteration J for boosting.
Output:
A strong classifier consisting of J weak classifiers.
Steps:
1. Initialize the weight distribution $\omega_{1,i} = 1/2m, 1/2l$ for $y_i = -1, 1$, where m and l are the number of negative and positive samples. The total number of training samples $N = m + l$;
2. For $j = 1 : J$

2.1 Normalize the weights: $\omega_{j,i} = \omega_{j,i} / \sum_{i'=1}^{N} \omega_{j,i'}$;

2.2 For each weak classifier c_j, select 3 features as the tree node $h_{j,u}$ (u=1,2,3) to minimize the error rate e and maximize the decrease of error rate Δe, where $e = \sum_i \omega_{j,i} |h_{j,u}(\boldsymbol{x}_i) - y_i|/2$ and $\Delta e = e_q - e_{q+1}$ (q=1,2, e_q and e_{q+1} are the error rate of parent and child node);

2.3 For each leaf node $h'_{j,k}$ (k=1,2,3,4), a weight $\alpha_{j,k}$ is computed as $\alpha_{j,k}$=0.5log($\frac{1-e_{j,k}}{e_{j,k}}$), where $e_{j,k}$ is the error rate of the kth leaf node of the jth weak classifier. The larger $e_{j,k}$ is, the smaller α_k a leaf node is assigned with.

2.4 Update sample weights: $\omega_{j+1,i} = \omega_{j,i} \beta_j^{1-\epsilon_i}$, where $\epsilon_i = 1$ for misclassified samples and $\epsilon_i = 0$ otherwise, and $\beta_j = \frac{e_j}{1-e_j}$. Since $e_j < 0.5$, β_j is thus between 0 and 1. Therefore, $\omega_{j+1,i}$ becomes smaller for correctly classified samples while staying the same for misclassified ones;
3. The final strong classifier is:

$$H(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \sum_{j=1}^{J} c_j - t > 0 \\ 0 & \text{else} \end{cases} \tag{4}$$

where c_j represents the score of the jth weak classifier and is defined as $c_j = \alpha_{j,k} h'_{j,k}$ (k is the index of the leaf node that an input image is classified into); t is a threshold automatically selected based on the performance of the corresponding strong classifier that runs on the validation set. Specifically, the desired detection rate (DDR) for a strong classifier is manually set in advance. The threshold t is automatically selected so that the detection rate > DDR. Fig. 3(d) shows the strong classifier built up by decision trees.

b. Cascade AdaBoost

The detection time of a strong classifier is proportional to the number of weak classifiers it contains. So it usually takes an extremely long time when using a single strong classifier to detect every sub-image. In order to speed up the detection, in our method, a cascade classifier is constructed. Each stage of the cascade is a strong classifier. Smaller boosted classifiers are constructed at early stages to remove most negative sub-images that can be easily rejected while detecting almost all positive ones. Then, more complex classifiers are deployed later to remove the false positives difficult to reject. Thus, a cascade of strong classifiers is constructed by gradually adding more complex strong classifiers at subsequent stages, as is shown in Fig. 4(e).

In order to improve the performance of the later strong classifiers, the training samples are iteratively updated during the construction. Specifically, positive samples remain the same for each stage, while negative samples for the subsequent stage are collected from the false positive sub-windows detected by the current detector that runs on IVOCT images.

During testing, an input sub-image goes through the cascade classifier stage by stage until rejected by one stage ($H_s(\boldsymbol{x}) = 0$), or successfully passing through all stages ($H_s(\boldsymbol{x}) = 1$ for all stages). If it passes through the whole cascade classifier, it is usually counted as a true positive. However, some false positives may still exist. In order to remove as much false positives as possible while retaining a relatively high detection rate, a score S_s for the sth strong classifier is computed as:

$$S_s = \sum_{j=1}^{J} c_{s,j} - t_s \tag{5}$$

Here, $c_{s,j}$ is the jth weak classifiers of the sth strong classifier, and t_s is the threshold of the sth strong classifier, as is mentioned in Eq. (4). Suppose that the cascade classifier has a total of N_s stages, the sum $\sum_{s=1}^{N_s} S_s$ can be seen as a score which measures how confident the cascade classifier is that the input sub-image is a positive one. A threshold T is set so that a sub-image is counted as a true positive when:

$$\begin{gathered} H_s(\boldsymbol{x}) = 1, s = 1, 2 \ldots N_s \\ \sum_{s=1}^{N_s} S_s - T > 0 \end{gathered} \tag{6}$$

With the cascade classifier, struts detection can be automatically conducted in IVOCT images.

2.1.3. Detection

Fig. 4 demonstrates the detailed process for struts detection in IVOCT images. There are three main steps in total: pre-process, detection, and post-process.

Firstly, the input IVOCT image is preprocessed to segment the lumen, guide wire and protective sheath to determine a detection region, and then construct a series of multiscale images for detection, as is shown in Fig. 4(b) to (c). In our method, Dynamic Programming (DP) is applied for the segmentation. Details can be seen in our previous work [20, 21]. The lumen is then expanded by a value ε to ensure that all struts are within the expanded lumen contour. The regions outside the expanded lumen, or within the guide wire and protective sheath, are removed to construct the detection region, as Fig. 4(b) shows. Besides, since the size of BVS struts varies, the image is down sampled into a series of multiscale images based on a scale factor $\sigma = [1, 1.5, 1.5^2, 1.5^3, 1.5^4]$, as Fig. 4(c) shows.

The next step is to detect the position and size of BVS struts by the trained cascade classifier. A sliding window of fixed size $L \times L$ ($L = 24$ pixels) moves throughout each of the multiscale

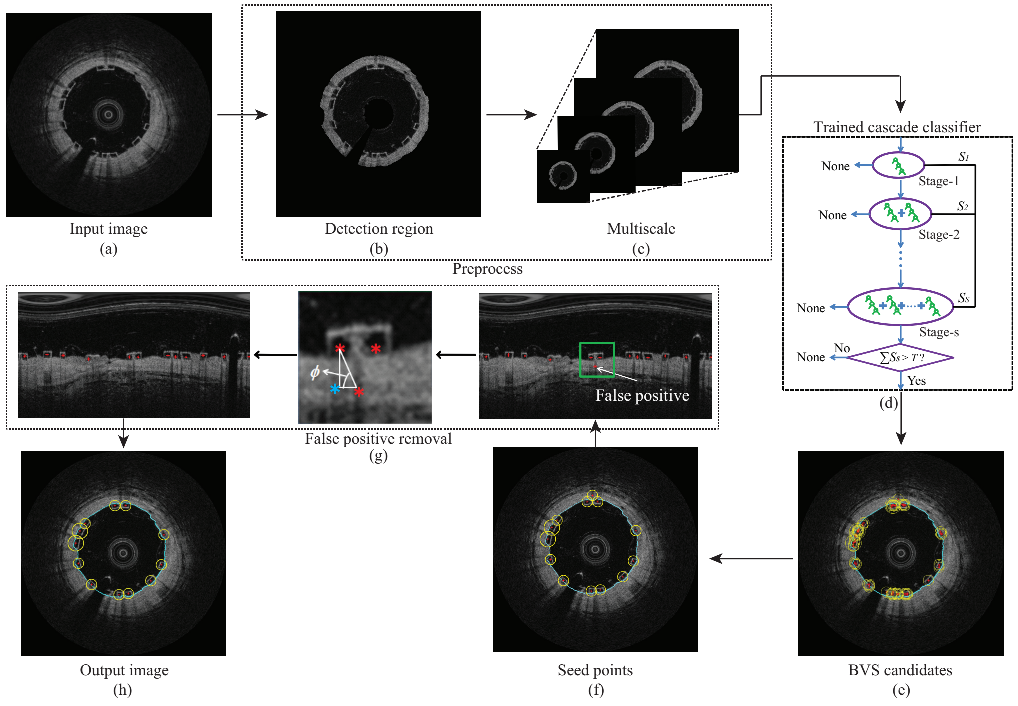


Fig. 4. The workflow for BVS struts detection with trained cascade classifiers.

images with a step of $K = 3$ pixels, and the cascade classifier determines whether the sub-window contains a BVS strut. If successfully detected as a candidate strut, the position of the strut is represented by the center point of the corresponding sliding window. Besides, since the strut varies in size from one to another, it's vital to find a local ROI for each strut so that it can be further segmented. In our method, the ROI is represented by a circle whose radius R is defined as: $R = \sigma \times L/2$, where σ is the scale factor and L is the size of the sliding window. For a large strut, it is usually detected at a large scale σ, and the ROI is therefore larger based on the definition. As Fig. 4(e) shows, the yellow circles represent ROIs and the red dots refer to candidate seed points.

Since a strut is detected at multi-scales, and the step K is far smaller than strut size, there is high chance that a strut is detected by more than one candidate seed point. Therefore it is important to cluster the candidates and select one that best describes the corresponding strut. Besides, some obvious false positives may occur after detection, and need to be removed. Therefore, post-process is conducted to optimize the result. It mainly contains two parts: candidates selection and false positive removal, as is shown in Fig. 4(f) to (h). Details are discussed as follows.

The main strategy for candidates selection is to search for a candidate seed point with the highest score $\sum S_s$ within a neighborhood. Here, we use a distance threshold $\lambda = 20$ pixels to determine the region of the neighborhood. In this way, for each detected strut, a seed point is found, and the corresponding ROI is determined to represent the position and size of the strut. The result of candidates selection is shown in Fig. 4(f).

On the other hand, during the imaging process, some light is absorbed by BVS struts and leaves some shadow artifacts in the same direction, which may lead to obvious false positives. Fig. 4(g) shows a false positive (blue dot) caused by shadow artifacts. These false positives need to be removed. The main strategy for false positive removal is shown in Fig. 4(g). The key idea is to transform the image into polar coordinate system and compute an angle ϕ for each seed point. Here, ϕ is the largest included angle of horizontal line and the line between two seed points.

Biomedical Optics EXPRESS

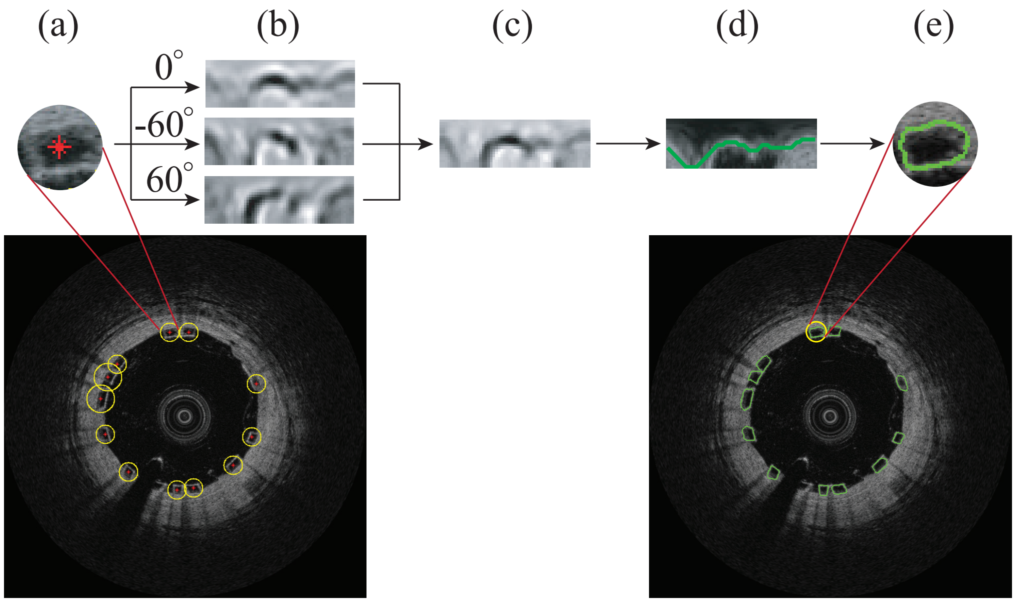


Fig. 5. The process of struts segmentation.

Specifically, the larger ϕ is, the more likely that the two seed points are within the same vertical line, and the lower one is therefore more likely to be a false positive. An angle threshold $\Phi = 60°$ is set in advance. If $\phi > \Phi$, it is counted as a false positive and removed. The remaining seed points are then transformed back into Cartesian coordinate system and are seen as the final result of struts detection, as is shown in Fig. 4(h).

2.2. Struts segmentation

In order to quantitatively measure the distance between struts and lumen for malapposition analysis, struts boundaries need to be segmented. Based on the detection, the segmentation problem can be resolved by traditional methods [8]. In our study, DP algorithm is applied to segment the detected struts. Fig. 5 shows the main steps of struts segmentation.

Firstly, each ROI, as Fig. 5(a) shows, is transformed into polar coordinate system characterized by angle γ and depth d. Details for the transformation can be found in the work [22]. Then, the transformed image I is filtered by 3 detectors that are respectively sensitive to edges of 0°, −60° and 60°. The 3 filtered images are shown in Fig. 5(b). After that, an energy image E is generated through a combination of the 3 filtered images. Suppose that the size the transformed image is $M \times R$, i.e., $\gamma \in [1, M]$ and $d \in [1, R]$, E can be defined as:

$$E(\gamma, d) = \min\{E_1(\gamma, d), E_2(\gamma, d), E_3(\gamma, d)\}, \qquad \gamma \in [1, M], d \in [1, R] \tag{7}$$

where $E_1(\gamma, d)$, $E_2(\gamma, d)$ and $E_3(\gamma, d)$ are the pixel values of the 3 filtered images. Fig. 5(c) visualizes the energy image. Our goal of struts boundary segmentation can be defined as searching for a path from column 1 to column M in the energy image E with the minimal accumulated pixel value C. Considering the connectivity constraint between adjacent columns on the strut boundary, this problem can be broken into a series of subproblems and resolved iteratively through the subproblems by DP algorithm. Specifically, the accumulated pixel value C in our method is defined as:

$$\begin{aligned} C(\gamma, d) &= E(\gamma, d), & \gamma = 1 \\ C(\gamma, d) &= \min C(\gamma - 1, d^*) + E(\gamma, d), & 1 < \gamma \le M \end{aligned} \tag{8}$$

where $C(\gamma, d)$ is the minimal accumulated pixel value from column 1 to point (γ, d), $d^* \in [d - \delta, d + \delta]$, and δ is a neighborhood coefficient that determines the connectivity between adjacent columns. In our method, $\delta = 1$ pixel. $E(\gamma, d)$ is the pixel value at point (γ, d) in the energy image E. In this way, a new matrix C is computed. By back tracking the minimal pixel value in C, the optimal path can be found. Fig. 5(d) shows the strut boundary segmented by DP algorithm in the polar coordinate system. The image, along with the segmented boundary, is then transformed back into Cartesian coordinate system, as is shown in Fig. 5(e).

2.3. Malapposition computation

After having obtained the segmentation data, we are able to compute the struts malapposition automatically based on the lumen and struts segmentation results. The minimum distance between a segmented strut and the lumen is computed. If the distance is larger than $150\mu m$, the strut is counted as a malapposed strut. Otherwise, it is an apposed strut. Details can be seen in Fig. 6(d)(e)(f).

3. Experimental setup

3.1. Materials

In our experiment, all scaffolds were the ABSORB 1.1 bioresorbable vascular scaffold (Abbott Vascular, Santa Clara, CA, USA). All IVOCT images were acquired using the FD-OCT system (C7-XR system, St. Jude, St. Paul, Minnesota). The resolution of the system was $10\mu m/pixel$. Each image has 704×704 pixels. A total of 15 baseline pullbacks, each of which contained 271 IVOCT images, were selected for the experiment. There were 12550 struts in the 15 pullbacks and all struts were manually segmented as the ground truth by an expert. Among the 15 pullbacks, 4 sets were used for training, another 4 sets were applied for validation, and the last 7 sets were employed for testing. However, limited by the computer memory, the training and validation data could not be fully used. Therefore, we randomly selected 1500 struts from training data and 2000 struts from validation data for training and validation, respectively. The 7 testing pullbacks were fully used, among which NO.2 and NO.3 contained both apposed and malapposed struts with blood clearly flushed, and NO.4 to NO.7 mainly contained apposed struts with blood artifacts.

3.2. Parameter settings

The thickness of a commonly seen BVS strut was about $150\mu m$ [23], and the length ranged from $150\mu m$ to $800\mu m$. Therefore, the size of the training samples was set to 24×24 pixels ($240 \times 240\mu m^2$), which was about 1.5 times the thickness of a common strut, to completely contain the strut without involving too much background. For struts larger or smaller than the common size, the sampling window should be 1.5 times the size of the strut and then normalized to 24×24 pixels. The scale factor was set to $\sigma = [1, 1.5, 1.5^2, 1.5^3, 1.5^4]$ to ensure that the largest struts can also be completely contained within a sliding window. The numbers of positive and negative training samples for each strong classifier were: $l = 1500, m = 4500$. The desired detection rate of a strong classifier was: $DDR = 99\%$.

For the parameters used in the detection and segmentation part, lumen expansion ε was set to 30 pixels, which was twice the width of a BVS strut (15 pixels), to ensure that all struts were within the detection ROI. The size L of the sliding window was the same as that of the training sample (24×24 pixels). Window step K was set to 3 pixels to ensure that every sub-window was detected. The distance threshold for candidates clustering was $\lambda = 20$ pixels, which was one quarter of the length of the largest strut.

3.3. Structure of the cascade classifier

Since AdaBoost rarely overfits in the low noise regime [24], there is no critical principle for the selection of the number of stages N_s and the number of weak classifiers J per stage. In our task, we empirically set N_s to 20, so that there are both enough simple classifiers at early stages and enough complicated classifiers at late stages. The number of weak classifiers J per stage was set through a trial and error process in which J was gradually increased until more than 99% positive samples were passed while a certain percentage of negative samples were rejected. The rejection percentage was set to 50% for the first stage, and was increased stage by stage until reaching 99% at the last stage. During training, the classification error on validation set was recorded to show whether the classifier was over trained. In this way, we finally trained a 20-stage cascade classifier with a total of 1326 weak classifiers. The first 5 stages contained 1 to 5 weak classifiers each, followed by stage 6 to stage 10 which contained 10 to 30 weak classifiers. Stage 11 to stage 20 consisted of more complex classifiers which contained 60 to 150 weak classifiers. It took about 84 hours in total to train the cascade classifier, and the classification error converged to around 3%, indicating that the classifier was not over trained.

3.4. Evaluation criteria

To quantitatively evaluate the performance of our method, true positive rate (TPR), false positive rate (FPR) and F measure were computed for the testing pullbacks and were defined as:

$$\begin{aligned} TPR &= TP/(TP+FN) \\ FPR &= FP/(TP+FP) \\ F &= 2\times(1-FPR)\times TPR/(1-FPR+TPR) \end{aligned} \tag{9}$$

where TP, FN and FP referred to the number of true positives, false negatives and false positives respectively.

For quantitative evaluation of detection, the center position error (CPE), defined as the distance between a seed point and the center of the corresponding ground truth, was calculated. In our experiment, CPE was computed by:

$$CPE = \sqrt{(x_s - x_g)^2 + (y_s - y_g)^2} \tag{10}$$

where (x_s, y_s) and (x_g, y_g) were the location of a seed point and the corresponding ground truth, respectively.

For quantitative evaluation of segmentation, Dice coefficient was applied to measure the coincidence degree of the area between the segmentation result and ground truth. In our experiment, Dice coefficient was defined as:

$$Dice = 2\times(A_s \cap A_g)/(A_s + A_g) \tag{11}$$

where A_s and A_g were the area of the segmentation result and ground truth, respectively.

For quantitative evaluation of malapposition analysis, malapposition rate was computed for both our method and ground truth. Here, malapposition rate was defined as the proportion of malaoppsed struts to all struts.

4. Results

4.1. Qualitative results

Fig. 6 shows some of the detection and segmentation results. White translucent masks are the ground truth marked by an expert. Blue dots refer to detected seed points, and yellow circles refer to the corresponding ROI. Green and red curves represent apposed and malapposed BVS struts,

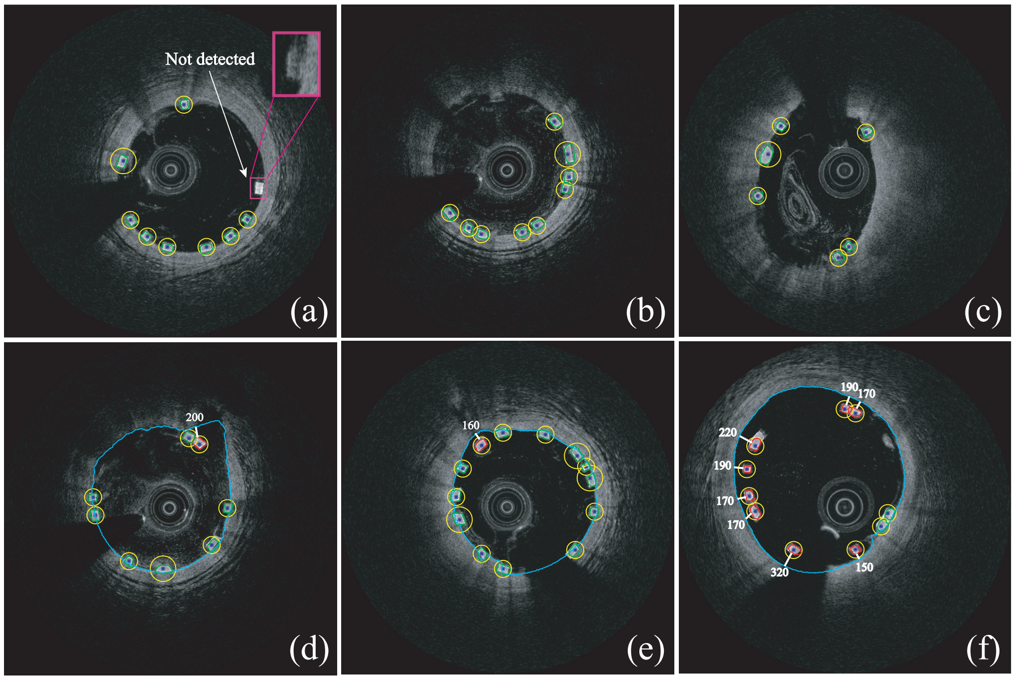


Fig. 6. The results of struts detection and segmentation in (a) normal images, (b)(c) images with severer artifacts and (d) to (f) images with apposed and malapposed struts. White translucent masks are the ground truth marked by an expert. Blue dots refer to detected seed points, and yellow circles refer to the corresponding ROI. Green and red curves represent apposed and malapposed BVS struts, respectively. For malapposed struts, the distance between lumen and strut (μm) is represented by the white line.

respectively. Fig. 6(a) is an IVOCT image that is commonly seen in a pullback. It can be seen that almost all struts were correctly detected and segmented, while no false positives remained. The only strut that failed to be detected was quite unclear in this frame, as the sub-image shows at the top right of Fig. 6(a). Shown in Fig. 6(b) and (c) are images with blood artifacts. Although blood artifacts were quite severer, our detector successfully detected all struts with no false positive detections. Fig. 6(d) to (f) are images that contain both apposed and malapposed struts, where cerulean curves represent lumen contour. For malapposed struts, the distance between struts and lumen were automatically labeled. It can be seen that for both images with or without blood artifacts, our method precisely detected and segmented almost all struts with few false positives. Besides, apposed and malapposed struts were correctly distinguished.

For accurate 3D visualization, Visualization Toolkit (VTK) is employed in our experiment. Fig. 7 shows the 3D reconstruction model of BVS struts within the coronary artery. Green and red curves refer to apposed and malapposed struts respectively, and yellow lumen were the coronary artery. It can be seen that apposed struts were embedded into artery lumen while malapposed struts were not. We can also see that the proximal of the BVS contained more malapposed struts than the distal. This is because the proximal of the coronary artery is usually larger than the distal, and is thus more difficult for struts to be completely apposed.

The above qualitative analysis suggested that our method was effective and robust for BVS struts detection and segmentation, and could be applied to evaluate and analyze struts malapposition in both 2D and 3D images.

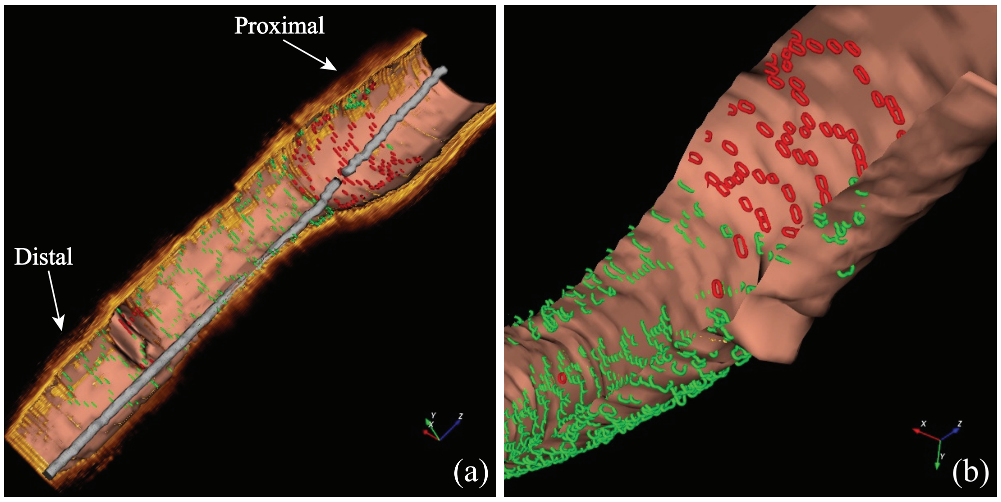


Fig. 7. The 3D reconstruction model of BVS struts within the coronary artery. Green and red curves refer to apposed and malapposed struts respectively. The golden layer is the tissue and the incarnadine layer inside is the coronary artery lumen. The Grey thin tube is the guide wire.

Table 1. The quantitative evaluation results.

			Detection				Segmentation	Malapposition Rate (%)	
Data set	No.F	No.GT	TPR	FPR	F	CPE	Dice	Manual	Automatic
NO.1	81	691	95.5	9.22	0.93	26.4	0.81	0.432	0.275
NO.2	119	928	91.3	10.2	0.91	31.2	0.81	1.71	2.01
NO.3	76	603	92.0	10.6	0.90	32.4	0.81	11.6	11.8
NO.4	118	1172	90.4	11.4	0.89	21.3	0.82	0	1.00
NO.5	78	604	93.4	16.8	0.88	24.8	0.82	0.166	0.191
NO.6	147	1188	90.8	15.4	0.88	30.0	0.82	0	0.705
NO.7	86	635	86.9	10.8	0.88	24.1	0.79	0	0
Average	-	-	91.5	12.1	0.90	27.2	0.81	-	-

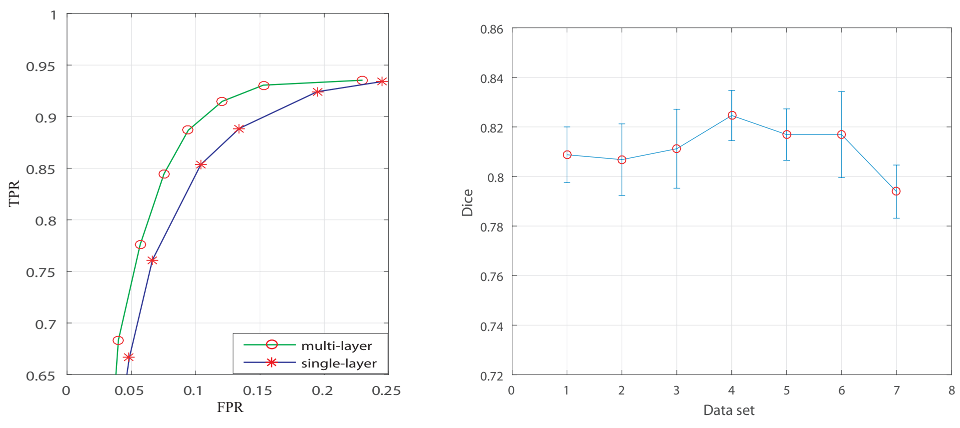
No.F: Number of frames evaluated; No.GT: Number of the ground truth; TPR: True positive rate (%); FPR: False positive rate (%); F: F-value; CPE: Center position error (µm).

4.2. Detection results

Table. 1 demonstrates the quantitative evaluation of struts detection, segmentation and malapposition analysis for each pullback.

For the evaluation of struts detection, a seed point is counted as a true positive if it is completely covered by a ground truth. Otherwise it is a false positive. It can be seen that our method reached a TPR of 91.5% and a FPR of 12.1% on average. By comparing the F measure between pullback NO.1-NO.3 (with no blood artifact) and NO.4-NO.7 (with severe blood artifact), we can see that the detection performance was, as expected, a little inferior when the image contained severe blood artifact. But the difference was quite little, and our method still successfully detected over 91% struts with average FPR less than 14% under severe blood artifact. Therefore, our method was robust for BVS struts detection under complex background. Besides, the average CPE was 27.2μm. Considering that the thickness of a BVS strut was usually up to 150μm, the CPE was relatively small, showing that our detection was accurate. The average F measure was 0.90, suggesting that the detection performance was outstanding.

Since the threshold T significantly influenced the detection performance, we manually changed



(a) (b)

Fig. 8. (a) The ROC curve of struts detection for multi layer tree-based classifier (green curve) and single layer stump-based classifier (blue curve). (b) The error bar of segmentation Dice coefficient for the seven test sets.

the value of T and plotted the ROC curve of struts detection, as is shown in Fig. 8(a), so that the best value of T could be found. The green curve was the ROC curve of our three layer tree-based classifier. It suggested that the best performance was achieved when FPR was about 12% and TPR was about 92%. At that very point, T=100. As a matter of fact, our detection and segmentation was conducted under the condition that T =100, and all the results were also obtained with T=100. Besides, in order to compare the performance of the single layer stump-based classifier and our multi layer tree-based classifier, we trained a single-layer classifier based on the same training samples and tested it on the same test sets. The blue curve in Fig. 8(a) shows the ROC curve of the single layer classifier. It can be seen that the green curve was higher than the blue one throughout the Fig., suggesting that our classifier performed better than the single-layer classifier. Since the best performance of the ROC curve was obtained at the best point that was nearest to the top left corner of the coordinate system, we compared the best performance of the two ROC curves at that point. The proposed multi-layer method reached a TPR=91.5% and FPR=12.1% at the best point, with F value at 0.90. In comparison, the single layer method achieved a TPR=85.4% and FPR=10.4% at the best point, with F value at 0.87. It suggested that the multi-layer method improved the F value by about 3.4% compared to the single layer method.

4.3. Segmentation results

To quantitatively evaluate the performance of struts segmentation, Dice coefficient was computed in our experiment. From Table. 1 we can see that the average Dice coefficient of the 7 pullbacks was 0.81, suggesting that our segmentation was accurate. To illustrate the robustness of struts segmentation, the error bar of Dice coefficient was plotted, as is shown in Fig. 8(b). It can be seen that the Dice coefficient kept stable at around 0.81 for all pullbacks, and the variance was no more than 0.02, indicating that our segmentation was robust.

4.4. Malapposition results

To quantitatively evaluate the performance of malapposition analysis, malapposition rate was computed for both our method and ground truth. It can be seen from Table. 1 that the overall malapposition rate for our method matched well with that of the ground truth, and sometimes a little higher. Recall rate and false positive rate of malapposed struts were also computed. The

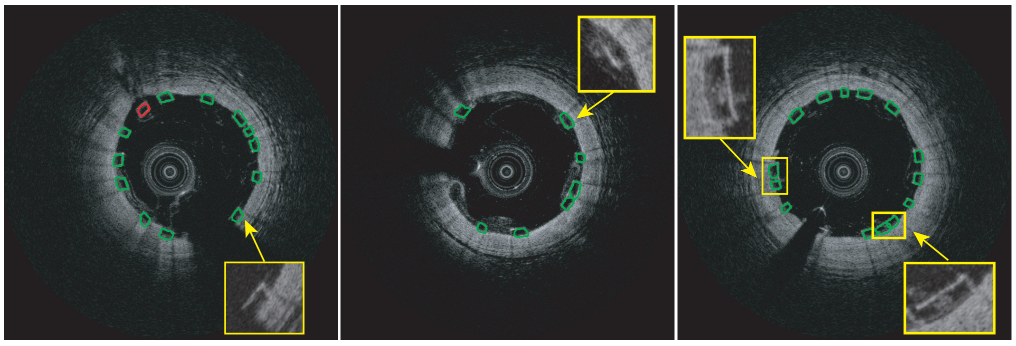

(a) (b) (c)

Fig. 9. The segmentation results of (a) partly hidden struts, (b)struts with unclear borders and, (c) struts with a bright spot in the center.

overall recall rate was 97.8% and the false positive rate was 26.7%. It suggested that our method successfully segmented almost all malapposed struts, but made some false positive detections as well. The false positive malapposed struts mainly came from the false positive detections such as blood artifacts during struts detection. This also explained why the malapposition rate of our method was generally a little higher than that of the ground truth.

4.5. Time consumption results

In our experiment, MATLAB 2015b was used for the realization of the algorithm. The computer configuration is: Intel Core i5-4460, CPU 3.20GHz. The key component of our algorithm was encapsulated as mex file and accelerated through C++. The average time consumption of struts detection and segmentation for a pullback was $14.81 sec$ in total, with $10.69 sec$ spent in detection and $4.12 sec$ in segmentation. As contrast, the traditional manual detection and segmentation for a pullback usually cost up to five hours by a human expert. Therefore, our method dramatically sped up malapposition analysis.

5. Discussion

The thresholds dramatically influenced the performance of our detection, segmentation and malapposition analysis. In our work, there were mainly two kinds of thresholds: weak classifiers thresholds θ, and the score threshold T. In our experiment, T was obtained based on the ROC curve tested on the validation set, and all θ were learned through training. In other words, the thresholds in our method were all automatically obtained by learning from a large number of samples. Generally speaking, the automatically learned thresholds were better than thresholds that were empirically set. Admittedly, the quantitative results of our method seemed inferior compared to Wang's work [14]. This might possibly result from the difference of testing data.

On the other hand, it is quite common in IVOCT images that a BVS strut is partly hidden, has unclear borders, or contains a bright spot in the center, which brings some difficulties to accurate strut boundary segmentation. Fig. 9 shows the segmentation results under these circumstances. Fig. 9(a) contains a struts that is partly hidden by the guide wire. Fig. 9(b) contains a struts whose boundary is blurred by surrounding background. Fig. 9(c) shows struts that contain bright spots in the center. It can be seen from Fig. 9(a) and (b) that our method accurately detected and segmented the struts that are partly hidden by the guide wire and struts with unclear borders. For struts that contain bright spots in the center, our method tend to detect the strut as two separated small struts, and conduct boundary segmentation for the two parts separately. It can be seen from Fig. 9(c) that, although the big strut was separated into two parts, the segmentation was accurate for both of the two parts. It indicates that our method was robust under different backgrounds.

6. Conclusion

In this paper, we proposed a novel framework for automatic malapposition analysis of BVS struts in IVOCT images. At first step, a cascade of classifiers is trained based on Haar-like features and AdaBoost algorithm and is then used for struts detection. At second step, DP algorithm was used to segment the struts boundaries. Based on the segmentation result, apposed and malapposed struts were distinguished automatically. The contribution of this paper mainly contains: 1) The proposal of a novel framework for automatic malapposition analysis that broke down the complex problem into two steps, which significantly simplified the task ; 2) The employment of Adaboost that trained a cascade of classifiers to obtain better robustness against blood artifacts; 3) The improvement of malapposition analysis velocity that reduced the analysis time for a pullback from 5 hours down to seconds. The qualitative and quantitative evaluation shows that our method is accurate and efficient for BVS struts detection and segmentation. The performance on images with or without blood artifacts were almost the same, suggesting that our method is robust under complex background. Besides, time consumption assessment shows that our method is fast enough for real-time malapposition analysis. It concludes that our method meets the clinical requirement of quantification and immediacy in malapposition analysis, and is of potential value in both clinical research and medical care. Since our method is currently applied to BVS (Abbott Vascular, Santa Clara, CA, USA) only with base-line data, future work will possibly extend to the analysis of metallic stent and other similar kinds of BRS in follow-up pullbacks and pullbacks with thrombus once the dataset is available. Besides, our current segmentation hasn't make use of the prior knowledge that a BVS strut was in box shape, future work will also focus on the improvement of segmentation accuracy based on the prior knowledge.

Funding

National Natural Science Foundation of China (Grant no. 81501542).

Disclosures

The authors declare that there are no conflicts of interest related to this article.